EEL 4712
Midterm 3 – Spring 2011
VERSION 1

Name: ________________________________

UFID: ________________________________

Sign your name here if you would like for your test to be returned in class:

____

IMPORTANT:
• Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.
• **As always, the best answer gets the most points.**

COVER SHEET:

Problem#:	Points
1 (6 points)	
2 (14 points)	
3 (6 points)	
4 (12 points)	
5 (6 points)	
6 (6 points)	
7 (6 points)	
8 (20 points)	
9 (24 points)	
10 (5 extra credit points)	

Total:

Regrade Info:

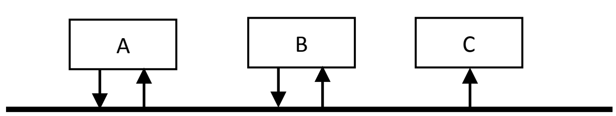
```
ENTITY _entity_name IS
PORT(__input_name, __input_name : IN STD_LOGIC;
__input_vector_name : IN STD_LOGIC_VECTOR(__high downto __low);
__bidir_name, __bidir_name : INOUT STD_LOGIC;
__output_name, __output_name : OUT STD_LOGIC);
END __entity_name;

ARCHITECTURE a OF __entity_name IS
SIGNAL __signal_name : STD_LOGIC;
BEGIN
-- Process Statement
-- Concurrent Signal Assignment
-- Conditional Signal Assignment
-- Selected Signal Assignment
-- Component Instantiation Statement
END a;

__instance_name: __component_name PORT MAP (__component_port => __connect_port,
__component_port => __connect_port);

WITH __expression SELECT
__signal <= __expression WHEN __constant_value,
__expression WHEN __constant_value,
__expression WHEN __constant_value,
__expression WHEN __constant_value;

__signal <= __expression WHEN __boolean_expression ELSE
__expression WHEN __boolean_expression ELSE
__expression;

IF __expression THEN
__statement;
__statement;
ELSIF __expression THEN
__statement;
__statement;
ELSE
__statement;
__statement;
END IF;

CASE __expression IS
WHEN __constant_value =>
__statement;
__statement;
WHEN __constant_value =>
__statement;
__statement;
WHEN OTHERS =>
__statement;
__statement;
END CASE;

<generate_label>: FOR <loop_id> IN <range> GENERATE
-- Concurrent Statement(s)
END GENERATE;
```

1) (6 points) Explain how a return instruction for the Small8 determines the return address of a function call.

2) (14 points) Write the assembly code to implement the following code. You only need to show the actual instructions, not the entire assembly program (e.g. lab test cases). Make sure to list any assumptions.

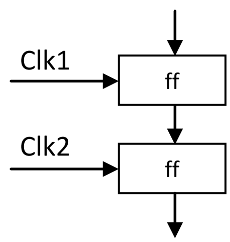
```
i=0;
while (i < 100) {
     i++;
}
```

3) (6 points) Draw a schematic of the FPGA circuit that would be synthesized for the following bus structure.



4) (12 points) List each step of an opcode fetch including all 8-bit, internal bus transfers. If you made significant changes to the architecture, make sure to describe them, but you must explain the purpose of each architectural register that is part of the fetch.

5) (6 points) Fix the metastability problems in the following circuit:



6) (6 points) What is the main limitation of a mux-recirculation synchronizer and how does a handshake address this limitation?

7) (6 points) Which of the following explains why a dual-flop synchronizer cannot be used to synchronize multiple-bit signals?

(a) Multiple bits take longer to stabilize
(b) Different bits may stabilize to different values
(c) Routing delays between clock domains may cause each bit to arrive in different cycles in the destination domain
(d) b and c
(e) a, b, and, c

8) (20 points) Create a memory initialization file for the following assembly code. Add a comment at the beginning of each instruction. *You will likely need to break your answer up into two columns to fit on the page.*

```
INPORT0       EQU    $FFFE
OUTPORT0      EQU    $FFFE
MASK          EQU    $0F

BEGIN:
       LDAA   INPORT0
       STAR   D
       LDAI   MASK
       ANDR   D
       STAR   D
       LDAA   VALUE1
       CLCR
       ADCR   D
       STAA   OUTPORT0
INFINITE_LOOP:
       CLRC
       BCCA   INFINITE_LOOP

* Data Area
VALUE1:       dc.b   $55

       END    BEGIN
```

```
Depth = 256;
Width = 8;
Address_radix = hex;
Data_radix = hex;
% Program RAM Data %
Content
  Begin
```

```
[       ..00FF] : 00;
End;
```

9) a. (6 points) For the following pseudo-code and schedule for the body of the loop, what is the *minimum* number of adders and multipliers required by the datapath to execute this schedule, assuming each state corresponds to a different cycle? *Do not change the schedule (it is purposely suboptimal).*

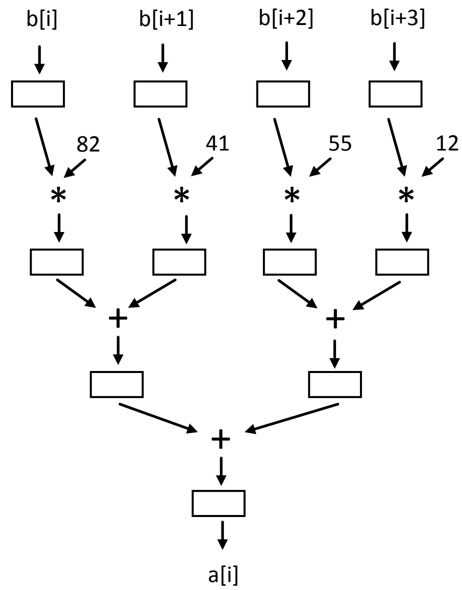
```
for (i=0; i < 1000000; i++) {
       a[i] = b[i]*82 + b[i+1]*41 + b[i+2]*55 + b[i+3]*12;
}
```

1. Load b[i], Load [i+1]
2. Load b[i+2], Load b[i+3], Multiply1 (b[i]*82), Multiply2 (b[i+1]*41)
3. Multiply3 (b[i+2] *55), Multiply4 (b[i+3]*12)
4. Add1 (Multiply1 + Multiply2), Add2 (Multiply3 + Multiply4),
5. Add3 (Add1+Add2)
6. Store a[i]

b. (6 points) Optimize the schedule to reduce the required number of adders by 1, without affecting the total number of cycles. You only need to specify the changes.

c. (6 points) Estimate the performance of the circuit (in # of cycles) with the schedule shown in part a.

d. (6 points) For the following pipelined datapath, what is the approximate performance of the loop in terms of number of cycles? Ignore initialization states. Assume memory bandwidth is sufficient to handle all datapath inputs/outputs.



10) (5 **extra credit** points) Shift registers are commonly used as delays in pipelined circuits. For deep pipelines, these shift registers can sometimes have a length of greater than 100, which requires many flip-flops in the FPGA. For long shift registers, a more efficient implementation is possible using RAMs. Explain how a block RAM and other additional logic can be used to implement a shift register.